Remarks

Applicants respectfully request reconsideration of the present application in view of the foregoing amendments and the following remarks. Claims 7, 14, 15, 21, 24-37, and 41-55 are pending in the application. Claims 7, 14, 15, 21, 24-37, and 41-55 are rejected. No claims have been allowed. Claims 7, 24, 30, 34, and 53 are independent. Claims 1-6, 8-13, 16-20, 22, 23, 25, 26, 35-40, and 47-52 have been canceled without prejudice. Claims 56-60 are new.

Cited Art

The Action cites Evans et al., "Splint Manual, Version 3.1.1-1," June 5, 2003 ("Splint Manual").

Claim Rejections under 35 USC § 102

The Action rejects claims 7, 14, 15, 21, 24-37, and 41-55 under 35 U.S.C. § 102(a) as allegedly being anticipated by Evans et al., "Splint Manual, Version 3.1.1-1." The Splint manual is dated "5 June 2003." Applicants do not admit that this Splint manual is prior art to the present application and reserve the right to provide evidence of prior conception.

For a rejection under 35 U.S.C. § 102 to be proper, the applied art must show each and every element as set forth in a claim. (*See* MPEP § 2131.) Applicants respectfully submit that the claims in their present form are allowable over the applied art because it does not teach or suggest all the claim limitations of the claims.

Applicants respectfully traverse these rejections and submit that the claims in their present form are allowable over the applied art.

Independent Claim 7

As amended, independent claim 7 recites in part: "*wherein the annotations on the first pointer are placed in an argument list to a function call that uses the first pointer as a parameter.*"

The applied art does not teach or suggest the above-cited language of amended independent claim 7. In particular, the applied art does not teach or suggest *wherein the annotations on the first pointer are placed in an argument list to a function call that uses the first pointer as a parameter.*

This is described in the Specification at, e.g., page 17, lines 6-9, reproduced below:

> The *writableTo* and *readableTo* annotations are placed on the buffer pointer. For example, the annotation *writableTo(byteCount(10))* can be placed on the buffer pointer for the function interface foo(char * buf) in the following manner:
>
> foo(*writableTo(byteCount(10))* char* buf)

Claim 7 is allowable. Claims 14, 15 and 21 depend from claim 7 and are allowable for at least the reasons given above in support of claim 7. Therefore, the rejections of claims 7, 14, 15 and 21 under 35 U.S.C. § 102 should be withdrawn. Such action is respectfully requested.

Independent Claim 24

Amended independent claim 24 recites in part:

> wherein the annotation comprises a first instance of a keyword, the first instance of the keyword indicating that the first value satisfies all usability properties necessary to allow a first function to rely on the first value, wherein other instances of the keyword identical to the first instance are operable to indicate that other values having different respective value types satisfy all usability properties necessary to allow functions to rely on the respective other values, <u>wherein use of the keyword associates a pre-determined set of usability properties with a value type,</u> and wherein the usability properties depend on the value type. [Emphasis added.]

The Splint manual does not teach or suggest the above-cited language of amended independent claim 24. In particular, the applied art does not teach or suggest an annotation that comprises a first instance of a keyword that indicates the first value satisfies all usability properties necessary to allow a first function to rely on the first value, wherein other instances of the keyword identical to the first instance are operable to indicate that other values having different respective value types satisfy all usability properties necessary to allow functions to rely on the respective other values, *wherein use of the keyword associates a pre-determined set of usability properties with a value type, and wherein the usability properties depend on the value type.*

This is explained in the Specification, at, e.g., page 18 line 3 through page 20, line 12 with reference to the "valid" property.

Regarding claim 24, The Examiner rejects the previous version of the claim stating: "Specifically, section 7.5 of the Splint manual describes "requires' and 'ensures' clauses, where the keywords 'requires' and ensures' are usable in multiple type contexts." [Action, page 3.]

Applicants respectfully disagree that the Splint manual teaches or suggest the current elements of claim 24. At page 41, the Splint manual describes use of 'requires' and ensures' clauses. The 'requires' clause is used to specify "a predicate that must be true at a call site." [Splint manual, p. 41, Sec. 7.5.] "An ensures clause specifies a predicate that is true at a call site after the call returns." [*Id.*] The syntax for these two clauses is shown on page 49 of the Splint manual. The syntax takes the form of 'ensures/requires A relative-operator B;' e.g., "requires A = 5"; or "ensures maxSet(s1) >= maxRead(s2)." That is, these clauses allow <u>a user</u> to specify that certain equations must be true within the call site; they do not discuss predefined properties or value types. For example, none of the portions of the Splint equation ("e.g., "A" or "B") are a value type; rather, they're variables. Further, this equation does not define or reference a *predetermined set of usability properties.* Therefore, these clauses do not teach or suggest "wherein use of the keyword *associates a pre-determined set of usability properties with a value type"* and so also fail to teach or suggest the further limitations "and wherein the usability properties depend on the value type."

Claim 24 is allowable. Claims 25-29 depend from claim 24 and are allowable for at least the reasons given above in support of claim 24. Therefore, the rejection of claims 24-29 under 35 U.S.C. § 102 should be withdrawn. Such action is respectfully requested.

Independent Claim 30

As amended, independent claim 30 recites in part:

> inserting an annotation having an argument in the computer program code, wherein the annotation annotates a value having a first declared value type with a first set of <u>annotation-specific</u> usability properties;
> wherein the annotation overrides the first set of <u>annotation-specific</u> usability properties of the first declared value type and indicates that the value has <u>annotation-specific</u> usability properties that depend on <u>annotation-specific</u> properties of a second value type denoted by the argument of the annotation. [Emphasis added.]

This is described in the Specification, at, e.g., page 18 line 3 through page 20, line 12 with reference to the "valid" property, and at page 20 line 13 through page 21, line 9 with reference to the "typefix" property. For example, at page 20 the Application describes the "typefix" property and states, "the typefix property can be used to override the declared C/C++ type. The interpretation of *valid* for *the annotated value* is obtained from the type given by the typefix instead of the declared C/C++ type." [Emphasis added.] Table 10 on page 19 of the Specification gives the interpretation of the *valid* annotation for different value types. In contrast, the "alt" notation described in the Splint manual only indicates that a declaration may be one of several possible types, overriding the C language rule that limits the declaration to a single type. [*See* Splint manual at page 24, section 4.]

The Splint manual does not teach or suggest the above-cited language of independent claim 30. Regarding claim 30, the Examiner states, "As per claim 30, [Splint manual] discloses inserting an annotation . . . wherein the annotation indicates that the value has usability properties that depend on the properties of a second value type denoted by the argument of the annotation." [*See* Action at p. 10.] Specifically, the Examiner cites the "alt" notation. [*See id.*]

Applicants respectfully disagree that the Splint manual teaches or suggest the elements of claim 30. At page 57, the Splint manual does describe use of the notation "/*@alt <type>, +@*/ . . . to indicate that an alternate type may be used." The "alt" notation described in the Splint manual indicates that "a declaration may be one of several possible types" [Splint manual, page 24, Sec. 4.4, 2nd para]. For example, a variable *c* can be declared as having three potential types, such that an int value, a char value or an unsigned char value may all be assigned to the variable *c*. [*Id.*] However, there is no indication that the "alt" notation itself gives any unusual properties not already possessed by the types (e.g., int, char, unsigned) assigned to a variable. Rather, the variable("c") is altered to be able to hold multiple garden-variety types. In particular, the "alt" notation as described in the Splint manual does not teach or suggest an annotation that *overrides a first set of annotation-specific usability properties of a first declared value type*. Thus, the Splint manual also does not teach or suggest that the value has annotation specific usability properties that depend on the properties of *a second value type* denoted by the argument of the annotation.

Claim 30 is allowable. Claims 31-33 depend from claim 30 and are allowable for at least the reasons given above in support of claim 30. Therefore, the rejection of claims 30-33 under 35 U.S.C. § 102 should be withdrawn. Such action is respectfully requested.

Independent Claim 34

Amended independent claim 34 recites in part:

> adding an annotation to a pointer in the computer program code, wherein the annotation describes transferring buffer properties from a second pointer to the pointer; and including a location parameter with the annotation, wherein the location parameter describes the logical buffer pointed to by the pointer. [Emphasis added.]

This is described in the Specification at, e.g., page 17, lines 10-14, reproduced below:

> The *aliased* property is useful for transferring buffer properties from one pointer to another. *notAliased* is useful for guaranteeing that two buffers are not aliased (i.e., that two buffers do not overlap). The *aliased* property is described in Table 8 below.

Property	**Meaning**
aliased(location)	Annotates a buffer pointer and states that the pointer points into the same logical buffer as *location*. The pointers need not be equal.

Table 8: The *aliased* property

The Splint manual does not teach or suggest the amended claim 34 language, above. Claim 34 is allowable. Claims 35-37 depend from claim 34 and are allowable for at least the reasons given above in support of claim 34. Therefore, the rejection of claims 34-37 under 35 U.S.C. § 102 should be withdrawn. Such action is respectfully requested.

Independent Claim 53

As amended, independent claim 53 recites in part:

> reading at least one annotation having an argument from the annotated computer program code, wherein the at least one annotation annotates a value having a first declared value type with a first set of usability properties <u>used only in an annotation context</u>, and wherein the annotation overrides the first set of usability properties <u>used only in the annotation context</u> of the first declared value type and indicates a second set of usability properties <u>used only in the annotation context</u> for the value that depend on the second value type denoted by the argument of the annotation<u>, such that the second set of usability properties are used in the context of a second annotation rather than the first set of usability properties</u>… [Emphasis added.]

This is supported in the Specification, at, e.g., page 20 line 13 through page 21, line 9 with reference to the "typefix" property, and at page 19, table 10, discussing the interpretations of the "valid" property for different value types.

The Splint manual does not teach or suggest the above-cited language of independent claim 53. Regarding claim 53, the Examiner states, "see, e.g., pp. 24 and 57, describing the alt annotation; the alt annotation overrides the type checking by adding alternative types, which incorporates the type checking associated with the newly added types." [*See* Action at p. 10.] The Examiner cites pages 24 57 of the Splint manual and, specifically, the "alt" notation. [*See id.*]

Applicants respectfully disagree that the Splint manual teaches or suggest the elements of claim 53. At page 57, the Splint manual does describe use of the notation "/*@alt <type>, +@*/ . . . to indicate that an alternate type may be used." The "alt" notation described in the Splint manual indicates that "a declaration may be one of several possible types" [Splint manual, page 24, Sec. 4.4, 2nd para] but there is no indication that there is anything unusual about these possible types, let alone that there are annotation specific usability properties associated with use of the "alt" tag. In particular, the "alt" notation described in the Splint manual does not teach or suggest "the at least one annotation annotates a value having a first declared value type with a first set of usability properties <u>used only in an annotation context</u>."

For example, at page 20 the Application describes the "typefix" property and states, "the typefix property can be used to override the declared C/C++ type. The interpretation of *valid* for *the annotated value* is obtained from the type given by the typefix instead of the declared C/C++ type." [Emphasis added.] Table 10 on page 19 of the Specification gives the interpretation of the *valid* annotation for different value types. In contrast, the "alt" notation described in the Splint manual only indicates that a declaration may be one of several possible types, overriding the C language rule that limits the declaration to a single type. [*See* Splint manual at page 24, section 4.]

Claim 53 is allowable. Claims 54-55 depend from claim 53 and are allowable for at least the reasons given above in support of claim 53. Therefore, the rejection of claims 54-55 under 35 U.S.C. § 102 should be withdrawn. Such action is respectfully requested.

Support for the Amendments

Support for the Amendments can be found in the specification and figures as originally filed. Further, the following specific examples are given:

Specification, page 18, line 3 to page 19, line 15.

Specification, page 21, line 21 to page 22 line 3;

Specification, page 13, line 4 to page 14, line 17, Tables 7A-7C, Table 9, Table 10.

Interview Request

If the claims are not found by the Examiner to be allowable, the Examiner is requested to call the undersigned attorney to set up an interview to discuss this application.

Conclusion

The claims in their present form should be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

By /Genie Lyons/
Genie Lyons
Registration No. 43,841